An Introductory Guide to the 1130

Santo D. Pratico

TABLE OF CONTENTS

The 1130 Computing System is an operator-oriented small scientific system which can also handle supporting commercial data processing applications.

The compact 1130 System features the 1131 CPU with core storage capacity of 4, 8, 16, or 32K 16-bit words. An additional 512,000 words of storage are available on-line with the disk storage feature of the 1131 Model 2. The interchangeable cartridge allows virtually unlimited off-line storage capacity. The CPU also includes a console with data displays and switches for operator control, a keyboard for data entry, and a console printer.

The basic 1130 System consists of the CPU and either the 1442 Card Read-Punch or 1134 Paper Tape Reader and the 1055 Paper Tape Punch. To either of these configurations can be added: disk storage, additional core storage, a 1627 Plotter, and the 1132 Printer.

Component Speeds

1131 CPU - 3.6 microseconds (access)
Disk Storage - 28.6 microseconds per word (transmission)
1442 Reader/Punch
Model 6 - Reads 300 cpm, Punches 80 cols/sec
Model 7 - Reads 400 cpm, Punches 160 cols/sec
1132 Printer (120 character line) - 80/110 lpm

1130 Disk Monitor System

The 1130 Monitor is a disk-oriented system that allows the user to assemble, compile, and/or execute individual programs or a group of programs.

The Monitor System consists of five distinct but interdependent programs.

A. The Supervisor program provides the necessary control for the stacked-job concept. It reads and analyzes the monitor control cards, and transfers control to the proper program.

B. The Disk Utility Program (DUP) is a group of routines designed to assist the user in storing information (data and programs) on the disk, and retrieving and using the information stored.

C. The assembler program converts user-writen symbolic-language source programs into machine language object programs.

D. The FORTRAN compiler converts user-written FORTRAN language source programs into machine-language object programs.

E. The Subroutine Library contains subroutines for data input/output, data conversion, and arithmetic functions.

The Monitor System coordinates program operations by establishing a communications area in memory which is used by various programs that make up the Monitor System. The complete Monitor System resides on disk storage. Only those routines or programs required at any one time are transferred to core storage for execution. This feature minimizes the core storage requirements and permits segmenting of long programs.

Programs can be stored directly on disk without the necessity of designating actual storage locations. The disk-stored programs and data are referred to by name when called for use. The Monitor, through the use of a table known as the Location Equivalence Table (LET) can locate any user program, subroutine, or file by a table search for the name. Stored with the name is the amount of disk storage (in disk blocks) required by the program or data.

Disk Storage Layout

Disk Storage is divided into three logical areas: IBM Systems area, User Storage Area, and Working Storage area.

A. IBM Systems Area

This area contains the integral parts of the 1130 Monitor, the Supervisor, the DUP, Fortran compiler, and the assembler. This area occupies 33 of the 200 cylinders of the disk.

B. User Storage Area (UA)

The area is used for storing IBM supplied subroutines, user-written programs, and data files. It also contains some system functions, namely, the LET, the Supervisor Control Record area, the Core Image Buffer, and sometimes a Fixed area with associated LET (FLET).

As each user-written program or data file is added to the User area, the space available for the Working Storage area decreases. Conversely, if a program is deleted, the Working Storage area increases by the amount of space the program formerly occupied in the User Area.

The user can also define optional area, called the Fixed area, which enables him to store programs and data files at fixed disk locations. The user can define the size of the Fixed area to be a whole number of cylinders (minimum of two) and increment the area by a whole number of cylinders at any time. Thus, programs or data files in this area can be referenced by absolute sector addresses, since they will not be moved. If a Fixed area is defined (via Disk Utility Program), it requires a LET of its own, known as FLET.

The Fixed area, when defined, occupies the beginning of the User area.

C. Working Storage Area (WS)

The Working Storage area is used for temporary storage. Most of the area is available to the user during execution of his programs. This area is also used by the Monitor, Assembler, and FORTRAN.

The disk storage on the 1130 is an enclosed single disk drive which revolves at 1500 rpm. It contains two surfaces and two read/write heads. There are 200 tracks per surface. A cylinder is defined

as an upper and lower track, therefore, the disk contains 200 two-track cylinders. Each track is divided into four sectors and each sector is defined as 321 data words, where the first word is a cylinder sector number. A sector represents the largest amount of data which can be moved with one instruction. The average sector read/write time is 30 milliseconds. Each sector can be subdivided into 16 disk blocks of 20 data words each. The length of programs or files in the LET is measured in terms of disk blocks.

In summary, the disk storage of the 1130 can be defined as:

1. 200 cylinders
2. 400 tracks
3. 1600 sectors
4. 25,600 disk blocks
5. 512,000 words

Disk Usage

Broadly speaking, the 1130 disk stores three types of information:

a. System programs
b. User programs
c. Data files

The first group of programs are not directly under the control of the user. The user can, however, at system generation time, delete the assembler or FORTRAN from his system, thus providing more room on the disk for general storage.

The second and third items are saved, modified, or deleted by the user's actions. These activities are accomplished through usage of the Disk Utility Program (DUP) which handles all disk maintenance operations.

DUP is a group of routines designed to accomplish the following:

1. Allocate disk storage as required by each program or data file to be stored. It is possible, for example, to compile and execute a program without permanently saving the program on disk.

2. Make these programs available in card or paper tape format.

3. Provide a printed status of the User, Fixed, or Working Storage areas.

4. Perform various disk maintenance operations.

The Disk Utility Program is called into operation by a // DUP control card. Once in control, DUP can accept any of twelve (12) control cards of its own. These cards provide for storing, deleting, dumping, and defining of disk storage areas. These cards, generally, contain the from and to symbols (PR, CD, PT, WS, UA, FX), program name, and optionally, a count, in addition to the function name.

For the sake of completeness, the twelve cards are listed below:

*	FUNCTION	FROM	TO	NAME	COUNT
(1)	(2-12)	(13-14)	(17-18)	(21-25)	(27-30)
	*DUMP	x	x	x	
	*DUMPDATA	x	x	x	x
	*STORE	x	x	x	
	*STORECI	x	x	x	x
	*STOREMOD	x	x	x	
	*STOREDATA	x	x	x	x
	*DELETE			x	
	*DUMPLET				
	*DWADR				
	*DEFINE FIXED AREA				x
	*DEFINE VOID ASSEMBLER				
	*DEFINE VOID FORTRAN				

The count field is in decimal, right justified. For data files, if the source is disk, this field specifies the number of sectors; if the source is cards, this field specifies the number of cards.

The following is a brief description of the above cards. The reader is directed to the Disk Monitor manual for complete details.

DUMP — This routine dumps information from FX, UA, or WS to cards, paper tape, or printer. It can also dump from UA or FX to WS.

DUMPDATA — This routine is exactly the same as DUMP, except only data is dumped. The number of sectors to be dumped must be specified in the count field.

STORE — This routine stores programs from cards, paper tape, or WS to the UA or WS. This routine is usually called after an assembly or compilation to save the program.

STORECI — This routine stores object programs from cards, paper tape, or WS to the UA or FX. The programs are converted to Disk Core Image format. The count field contains the count of *FILES records that are required for the program being stored.

STOREMOD — This routine moves a program or data from WS to UA or FX, overlaying on item specified by name in the UA or FX. This permits the user to modify an item in UA or FX without changing its name or relative position.

STOREDATA — This routine stores data from cards, WS, or paper tape to UA, FX, or WS. Each data file begins at the next available sector and the length is defined in numbers of sectors.

DELETE — This routine deletes a named program or data file from the UA or FX.

DUMPLET This routine dumps the contents of the Location Equivalence Table (LET) to the printer.

DWADR This routine writes sector addresses on every sector in WS.

DEFINE This routine defines variable parameters required by the Monitor System.

a. Define or increase the size of the Fixed area
b. Delete the assembler from the system
c. Delete FORTRAN from the system

Creating a Data File

Data files can be created in two ways; stored on disk from an external source (cards) or generated by an assembler or FORTRAN program.

The first method is relatively simple and an example should clarify the procedure.

```
// b JOB
// b DUP
*STOREDATA CD UA NAME COUNT
      -- Card File --
// b JOB
```

The count in this case is the actual number of cards following the control card. Once stored on disk this file can be accessed at any time by the name specified on the control card.

When data is program generated, the above procedure can be used with two exceptions. The designation CD would be replaced with WS and the count would indicate the number of sectors generated.

The user can easily calculate the number of sectors by dividing the number of words generated by 320. The deck setup follows:

```
// b JOB
// b XEQ NAMEA
      --Data cards, if any
// b DUP
*STOREDATA WS UA NAMEB COUNT
// b JOB
```

This setup will save a file generated by an assembler or FORTRAN program, however, the methods of generation are different.

The 1130 FORTRAN includes four statements for disk operations. These are DEFINE FILE, READ, WRITE, and FIND.

The DEFINE FILE statement specifies to the compiler the size and quantity of disk data records within files that will be used with a particular program and its associated subprograms. This statement is used only in mainline programs and its purpose is to divide the disk unit into files to be used by READ, WRITE, and FIND.

```
DEFINE FILE 1 (1000, 20, U, K)
```

The above statement would define file number 1 as containing 1000 20-word records. The letter U is mandatory. The variable name K is a count used in the FORTRAN program and it points to the next available record in the file.

The READ and WRITE statements direct the program to read or write the Kth record of file J.

```
READ (J'K) list
WRITE (J'K) list
```

After reading or writing a record the count, K, is automatically incremented by 1.

The FIND statement positions the read/write head to the Kth record of file J.

```
FIND (J'K)
```

The Monitor provides a method for using a created data file in a FORTRAN program, via the *FILES card. Once a data file is stored on disk and its name is known, it can be equated to a file number in the DEFINE FILE statement of a FORTRAN program. This card, which follows the // XEQ card, is formatted as follows:

```
*FILES (3, LEASE), (47, PURCH)
```

This would equate FORTRAN file numbers 3 and 47 with stored files LEASE and PURCH. The control cards to compile and execute a FORTRAN program using these files are:

```
// b JOB
// b FOR
     FORTRAN control cards
     FORTRAN source deck
// b DUP
*STORE    WS       UA    NAME
// b XEQ  NAME      1
*FILES (3, LEASE), (47, PURCH)
     Data cards, if any
// b JOB
```

The 1130 Assembler requires a more detailed procedure for disk operations and the following description will merely introduce the reader to the technique. A fuller description can be found in the Functional Characteristics manual (A26-5881).

In assembly language, all I/O is performed by using the XIO (Execute I/O) instruction. The operand of this instruction must address an IOCC (I/O Control Command) which details:

```
Address
Device
Function
Modifier
```

These fields of the IOCC define the location in core storage to be read into or written from, the device to be used, and the type of I/O operation desired.

Disk read and write instructions must be supplied the number of words to be transmitted, to a maximum of 321 words (one sector), and the actual sector number being read or written.

1130 Monitor Control Cards

Input to the Supervisor consists of one or more job decks, each preceded by a JOB Monitor control card, which defines the starting and ending points of the job.

When a Monitor control card is read, the system program required to do the subjob is read into core storage from disk storage. Every job is assumed to begin with no programs in Working Storage.

All monitor control cards have the following format:

Cols. 1-2: //
3: blank
4-7: Pseudo-operation code (left-justified)

The Monitor System recognizes the following nine (9) cards:

1.	// b *	comments
2.	// b JOB	- used to initialize a job sequence
3.	// b ASM	- read assembler into core for execution
4.	// b FOR	- read FORTRAN into core for execution
5.	// b PAUS	- halt until START is pressed
6.	// b TYP	- read succeeding monitor cards from keyboard
7.	// b TEND	- return to reading monitor cards from card reader
8.	// b DUP	- read Disk Utility Program into core
9.	// b XEQ	- read and transfer control to mainline program

Some of the above cards require more information besides the pseudo-op. Consult the Monitor manual (C26-3750) for particulars.

The Supervisor recognizes three (3) control cards of its own (LOCAL, NOCAL, FILES). These cards, when used, always follow the // XEQ card.

A. LOCAL (Load-on-call) -- This is an acronym denoting certain routines specified by the user to be loaded into a LOCAL overlay area as they are called. This technique is used when a program becomes too large to reside in memory in one core load and has to be segmented. The card itself names the mainline program and the subprograms to be classified as LOCAL's.

B. NOCAL (Load - Although - Not Called) -- subprograms are subprograms specified by the user to be included in the core load, even though they are not called. Examples of such a situation would be debugging aids or interrupt routines which the operator could branch to manually.

C. FILES -- this card is used to equate names of data files in the User Area to file numbers used in a FORTRAN program. This equation takes place at execution time.

The reader is reminded to consult the above-cited reference manual in using these three cards because there are certain rules that must be adhered to.

Examples of Deck Setups

A. To compile and execute a FORTRAN program:

```
// b JOB
// b FOR
     FORTRAN control cards
     FORTRAN source deck
// b DUP
*STORE       WS        UA        NAME
// b XEQ     NAME
     Data cards, if any
// JOB
```

B. To assemble and execute an ASSEMBLER program:

```
// b JOB
// b ASM
     Assembler control cards
     Assembler source deck
// b DUP
*STORE       WS        UA        NAME
// b XEQ     NAME
     Data cards, if any
// b JOB
```

C. To compile, assemble, include an object program and execute the mainline program:

```
// b JOB
// b ASM
     Assembler control cards
     Assembler source deck
// b DUP
*STORE       WS        UA          A
*STORECI     CD        UA          B
     Object program
// b FOR
     Fortran control cards
     Fortran source deck
// b DUP
*STORE       WS        UA          C
// b XEQ      C
     Data cards, if any
// b JOB
```

Full details on assembler and compiler control cards can be found in the previously cited reference manual. For the sake of completeness, these cards are listed below without explanation.

A. FORTRAN control cards

1. *NAME XXXXX
2. *IOCS (CARD, TYPEWRITER, KEYBOARD, 1132 PRINTER, DISK, PLOTTER)
3. **
4. *ONE WORD INTEGERS
5. *EXTENDED PRECISION
6. *ARITHMETIC TRACE
7. *TRANSFER TRACE
8. *LIST SOURCE PROGRAM
9. *LIST SUBPROGRAM NAMES
10. *LIST SYMBOL TABLE
11. *LIST ALL

B. ASSEMBLER

1. *TWO PASS MODE
2. *LIST
3. *LIST DECK
4. *LIST DECK E
5. *PRINT SYMBOL TABLE
6. *PUNCH SYMBOL TABLE
7. *SAVE SYMBOL TABLE
8. *SYSTEM SYMBOL TABLE
9. *LEVEL
10. *FILE
11. *COMMON

Appendix A - 1130 FORTRAN

1130 FORTRAN is very similar to most FORTRAN languages except for certain features. The following will serve to detail these exceptions.

1. The largest integer constant acceptable is 32767.

2. Real constants consist of 1-7 or 1-10 significant decimal digits, depending on the precision specified.

3. Variable names may consist of 1-5 alphameric characters. (Note- most compilers allow 6 characters.)

4. Arrays may have up to three subscripts.

5. The first specification statement encountered must define the size of an array.

6. Mixed mode arithmetic is allowed with computations calculated in the real mode.

7. The size of COMMON specified in the mainline must be at least as large as the largest COMMON specified in any subprograms.

8. The print line is 120 characters.

9. Hollerith information can be enclosed in apostrophes in a FORMAT statement.

10. Logical unit numbers
    - 1 - Console Printer
    - 2 - Cord Read/Punch
    - 3 - 1132 Printer
    - 6 - Keyboard

11. The following statements comprise the FORTRAN language on the 1130:

    a. GO TO
    b. GO TO (n1, n2, n3, ...), I
    c. IF (expression) n1, n2, n3
    d. DO
    e. CONTINUE
    f. STOP or STOP n
    g. PAUSE or PAUSE n
    h. END
    i. READ (a, b) list
    j. WRITE (a, b) list
    k. FORMAT
    l. REAL

m. INTEGER
n. EXTERNAL
o. DIMENSION
p. COMMON
q. EQUIVALENCE
r. FUNCTION
s. SUBROUTINE
t. RETURN
u. CALL
v. DEFINE FILE
w. READ (a'b)
x. WRITE (a'b)
y. FIND (a'b)

12. Version 2 Improvements (due year-end 1967)

a. DATA statement
b. BACKSPACE statement
c. REWIND statement
d. END FILE statement
e. READ (a)
f. WRITE (a)
g. PDUMP routine
h. T-format code

Appendix B - Assembler

The 1130 Assembler contains the following 29 instructions.

LD	-	Load
LDD	-	Load Double
STO	-	Store
STD	-	Store Double
LDX	-	Load Index
STX	-	Store Index
STS	-	Store Status
LDS	-	Load Status
A	-	Add
AD	-	Add Double
S	-	Subtract
SD	-	Subtract Double
M	-	Multiply
D	-	Divide
AND	-	Logical AND
OR	-	Logical OR
EOR	-	Logical Exclusive OR
SLA	-	Shift Left ACC
SLT	-	Shift Left ACC and EXT
SLCA	-	Shift Left and Count ACC
SLC	-	Shift Left and Count ACC and EXT
SRA	-	Shift Right ACC
SRT	-	Shift Right ACC & EXT
RTE	-	Rotate Right ACC & EXT
BSC	-	Branch or Skip on Condition
BSI	-	Branch and Store Instruction Address Register
MDX	-	Modify Index and Skip
WAIT	-	Halt
XIO	-	Execute I/O

Appendix C - Manuals

3rd 68

Functional Characteristics	A26-5881-4
Assembler	C26-5927-4
FORTRAN	C26-5933
Subroutine Library	C26-5929-4
Disk Monitor System	~~C26-3750~~
Dick Monitor System - Version 2	~~C26-3709~~ C26-3717-1
Commercial Subroutine Package	H20-0241
Data Presentation System	
Operator's Manual	H20-0337
System Manual	Y20-0089
User's Manual	H20-0338
Utility Routines	C26-5931
Scientific Subroutine Package	H20-0252
Statistical System	
User's Manual	H20-0333
System Manual	Y20-0093

Appendix D - Coding Examples

The following five coding forms will illustrate:

a. Monitor control cards and field position.

b. DUP control cards with appropriate card columns.

c. A typical FORTRAN job setup.

d. A typical Assembler job setup.

e. A FORTRAN program which creates and prints a data file.

System		Punching Instructions		Sheet
Program		Graphic	Card Form# *	
Programmer	Date	Punch		

FIELD IDENTIFICATION

MONITOR CONTROL CARDS

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890
// *
// JOB T XXXXX
// ASM
// FOR
// PAUSE
// TYP
// TEND
// DUP
// XEQ XXXXX L YY Z

*LOCAL
*NOCAL
*FILES(XXXXX,YYYYY),(AAAAA,BBBBB)
```

1-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80
1234567890123456789012345678901234567890123456789012345678901234567890

System		Punching Instructions		Sheet
Program		Graphic	Card Form# *	
Programmer	Date	Punch		

FIELD IDENTIFICATION

DUP CONTROL CARDS

1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
// DUP							
*DUMP	FF TT	XXXXX					
*DUMPDATA	FF TT	XXXXX CCCC					
*STORE	FF TT	XXXXX					
*STORECI	FF TT	XXXXX FFFF					
*STOREDATA	FF TT	XXXXX CCCC					
*STOREMOD	WS TT	XXXXX					
*DUMPLET							
*DWADR							
*DELETE		XXXXX					
*DEFINE FIXED AREA		NNNN					
*DEFINE VOID ASSEMBLER							
*DEFINE VOID FORTRAN							

System		Punching Instructions		Sheet
Program		Graphic	Card Form# *	
Programmer	Date	Punch		

FIELD IDENTIFICATION

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
// JOB          SAMPLE FORTRAN SETUP
// DUP
*DELETE         SAMPL
// FOR
*IOCS(CARD,TYPEWRITER,KEYBOARD,1132 PRINTER,DISK)
*NAME SAMPL
*ONE WORD INTEGERS
*LIST ALL
                } FORTRAN SOURCE PROGRAM
// DUP
*STORE      WS UA SAMPL
// XEQ SAMPL  0
                } DATA CARDS, IF ANY
// JOB
```

System		Punching Instructions		Sheet
Program		Graphic	Card Form# *	
Programmer	Date	Punch		

FIELD IDENTIFICATION

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
// JOB          SAMPLE ASSEMBLER SETUP
// DUP
*DELETE         SAMPL
// ASM
*LIST
*PRINT SYMBOL TABLE
                ASSEMBLER SOURCE PROGRAM
// DUP
*STORE      WS  UA  SAMPL
// XEQ SAMPL   0
                DATA CARDS, IF ANY
// JOB
```

1-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80

System		Punching Instructions		Sheet
Program		Graphic	Card Form# *	
Programmer	Date	Punch		

FIELD IDENTIFICATION

FORTRAN PROGRAM: CARD → DISK → PRINTER

```
      DEFINE FILE 726(100,80,U,KOUNT)
      DIMENSION KARD(80)
C     INITIALIZE COUNTER TO RECORD POSITION 1
      KOUNT=1
C     READ 100 CARDS AND PLACE ON DISK
      DO 10 I=1,100
      READ(2,5) KARD
    5 FORMAT(80A1)
      WRITE(726'KOUNT) KARD
   10 CONTINUE
C     RE-INITIALIZE COUNTER TO RECORD POSITION 1
      KOUNT=1
C     READ RECORDS FROM DISK AND PRINT
      DO 20 I=1,100
      READ(726'KOUNT) KARD
      WRITE(3,15) KARD
   15 FORMAT(1X,80A1)
   20 CONTINUE
      STOP
      END
```